BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a data model which encapsulates data validation as well as data, according to the present invention;

Fig. 2 illustrates how the data model of Fig. 1 allows data and its validation to be easily and efficiently shared among multiple presentations;

Fig. 3 shows a sample data model validation object created according to an embodiment of the present invention;

Fig. 4 provides a flowchart depicting logic underlying an implementation of the present invention; and

Figs. 5A and 5B show sample code which may be used to perform validation of data, according to preferred embodiments of the present invention.

DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention defines novel techniques for performing data validation. Validation is coupled with, or encapsulated with, the data values to which the data validation pertains, thereby becoming a part of the data model itself. This approach enables real-time data validation, as a user interacts with a data model through an executing application or GUI window interface.

This data model is illustrated in Fig. 1. See generally element 100, in which data model 120 encapsulates data 110 and also a set of validation criteria 115. This data model 100 may interact with a GUI widget 105 which stores data in the model and/or retrieves stored data, typically when interacting with an end user. (Interaction is discussed further below with reference to Fig. 2 as well.) In object oriented programming terms, the coupling of the validation and the data forms a complex object.

The approach of the present invention offers a number of advantages over prior art validation techniques. As one advantage, because validation criteria (such as rules which describe acceptable data values and formats) can now float with the data model, and therefore with the data values. That is, the validation can travel with the data values, from one implementation to another, as if the validation was simple data. The object containing the data and its validation can be passed between applications, stored, and so forth, and the receiving application need not be aware that validation is contained within the object it is operating upon.

Another advantage of the present invention is that the validation process is handled at an early point, that is, when the data model is being populated. This is especially beneficial if the data model is being pre-loaded with data values, for example before presenting a GUI display of default values to a user. Validation may also be performed as an application mutates data values. In prior art approaches, validation typically does not occur until run-time.

A further advantage of the present invention is that separating the data validation from the

GUI allows for easily and efficiently sharing the data and data validation among multiple presentations. This is illustrated by Fig. 2. For a particular data model 100, either GUI widget 105 or GUI widget 205 might be used to display the contained data 110 to a user, and to validate any revised data values the user provides, according to the single shared validation object 115. For example, suppose data 110 contains a user's home phone number. GUI widget 105 might display this data value using a text field widget that allows modifying the existing phone number. GUI widget 205, on the other hand, might display this data value as a label in a radio button list, along with the user's work phone number as the label of another radio button. Rather than including validation rules regarding proper values for phone numbers, and proper formatting of phone numbers, in each widget as is typically done in prior art approaches, the widgets 105 and 205 rely on the data model 100 to handle these details. Similarly, if a given GUI window includes many different types of data, the window and its widgets can delegate responsibility for validating these various types of data to the data models. Because the data models are already designed to store a particular type of data, very little additional effort should be required for a designer to incorporate the validation directly into the data model once the teachings of the present invention are known. This approach also reduces the amount of code required for the GUI widgets, lowers their complexity, and increases their reusability.

Prior art approaches to validation typically tie the validation to a GUI. Separating validation from the GUI itself, according to the present invention, allows changes to a GUI to be implemented more easily and more quickly. The GUI developer can focus on the GUI layout and GUI widgets, and need not be concerned with the validation rules for the data and the effect of